MOTION IMAGERY STANDARDS BOARD	MISB ST 1608
STANDARD	
Transport of Motion Imagery and Metadata over GigE Vision	22 June 2017

1 Scope

This standard defines recommended data block transmission modes for carriage of Class 0 Motion Imagery, Class 2 Motion Imagery and KLV Metadata over GigE Vision.

The standard specifies use of MISB ST 1507 and MISB ST 1603 for timestamping Motion Imagery.

The standard defines the format and placement for KLV metadata within a GigE Vision interface.

2 References

[1] Advanced Vision+Imaging (AIA) GigE Vision 2.0, Video Streaming and Device Control Over Ethernet Standard, Apr 2013.

[2] MISB ST 0107.2 Bit and Byte Order for Metadata in Motion Imagery Files and Streams, Feb 2014.

[3] MISB ST 1507.1 Motion Imagery Sensor Timing Metadata, Jun 2017.

[4] MISB ST 1603.1 Time Transfer and Enhanced Precision Time Stamp Metadata, Feb 2017.

[5] MISB ST 0603.4 MISP Time System and Timestamps, Feb 2016.

[6] EMVA GenICam Standard Ver 3.0.1, May 2016.

[7] MISB MISP-2017.1: Motion Imagery Handbook, Oct 2016.

3 Revision History

Revision	Date	Summary of Changes
ST 1608	6/22/2017	• Initial release

4 Acronyms

GVCP	GigE Vision Control Protocol
GVSP	GigE Vision Streaming Protocol
KLV	Key Length Value

MISB	Motion Imagery Standards Board
MISP	Motion Imagery Standards Profile
SMPTE	Society of Motion Picture & Television Engineers

5 Introduction

GigE Vision is a global camera interface specification built on the Gigabit Ethernet communication protocol. GigE Vision was developed by a consortium of companies invested in the (machine) vision industry. It supports interfacing between a GigE Vision device and a network card using standard CAT-5e/6 cable, or any other physical medium supported by Ethernet.

GigE Vision is composed of two protocols: GigE Vision Control Protocol (GVCP) is an application layer protocol allowing an application to configure and control a device (typically a camera), and to instantiate stream channels (GVSP transmitters and receivers, when applicable); GigE Vision Streaming Protocol (GVSP) is an application layer protocol allowing a GVSP receiver to receive image data, image metadata or other information from a GVSP transmitter.

The GigE Vision Specification [1] mandates use of the User Datagram Protocol (UDP) with IPv4 and a fixed IP header size of 20 bytes (Section 23 in [1]). GigE Vision allows up to 512 channels for streaming data. A stream channel enables data transfer from a GVSP transmitter to a GVSP receiver. This is limited only by the device itself. Rules for channel number selection are defined in the GigE Vision Specification; the first stream channel must begin at index zero with no gaps sequentially.

GigE Vision supports several payload types. MISB ST 1608 details the use of four: Image, Chunk Data, JPEG 2000 and H.264. The Image payload type supports Class 0 Motion Imagery (i.e. uncompressed image data). The Chunk Data payload type supports tagged blocks of data, useful for streaming metadata only. The JPEG 2000 and H.264 payload types support Class 2 Motion Imagery (i.e. compressed image data). In the streaming of imagery, this standard assumes metadata accompanies the imagery. Motion Imagery and Metadata transport are discussed in Section 6; Metadata-only transport is discussed in Section 7.

6 Motion Imagery and Metadata Carriage

This section defines the GigE Vision payload type for the transport of non-compressed Class 0 Motion Imagery, and H.264-compressed and JPEG 2000-compressed Class 2 Motion Imagery, and the rules for the placement and recognition of KLV metadata.

Various use cases can take advantage of a multitude of stream channels. In a most basic configuration, one stream channel is invoked to deliver one stream of Class 0 Motion Imagery and metadata. Another use case may supply a H.264/AVC compressed Class 2 Motion Imagery and metadata stream in addition to its source Class 0 Motion Imagery in a parallel stream channel. A third example might be a region-of-interest derived from the source imagery, but sent in a separate stream channel.

The GigE Vision Specification facilitates flexibility in image array sizes, limiting height and width only by their 32-bit word sizes. Allowed pixel layouts are identified in the GigE Vision Specification Section 26.

For Motion Imagery/metadata transport, the GigE Vision Standard Transmission Mode is mandated.

Requirement	
ST 1608-01	Transmission of Motion Imagery with metadata on a stream channel shall use the Standard Transmission mode in accordance with the GigE Vision Specification.

In Standard Transmission mode, data is aggregated into data blocks, where each data block consists of a Data Leader Packet, a Data Payload Packet and a Data Trailer Packet as shown in Figure 1. MISB ST 1608 leverages the *image extended chunk mode* of GigE Vision, where image data and its corresponding metadata are contained within one data payload.

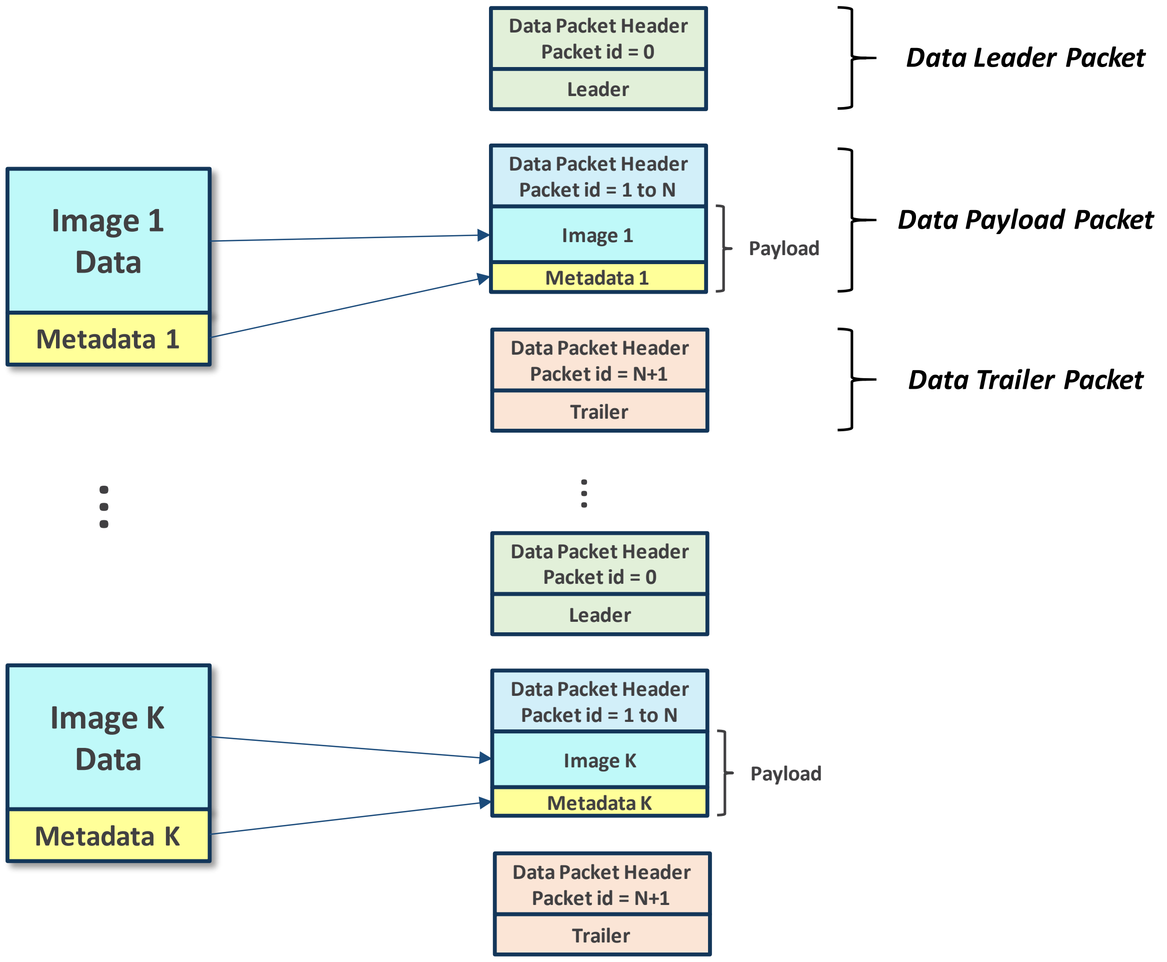


Figure 1: Image Data Mapped into Data Block - Standard Transmission Mode

The Data Leader Packet (composed of Data Packet Header and Leader) signals the beginning of a new data block. Following the Data Leader Packet is a Data Payload Packet (composed of Data Packet Header and Payload), which contains the image data and metadata for one image. Finally, a Data Trailer Packet (composed of a Data Packet Header and Trailer) signals the end of a data block. All GVSP packets share the same basic header (GVSP Packet Header), which among other entries includes status, the id of a block of data, the packet id, and packet format.

The GigE Vision specification requires each block of data with a specific data type to be followed by an identifier unique to that data block called the Chunk ID, or CID, and the length of the data block as shown in Figure 2.

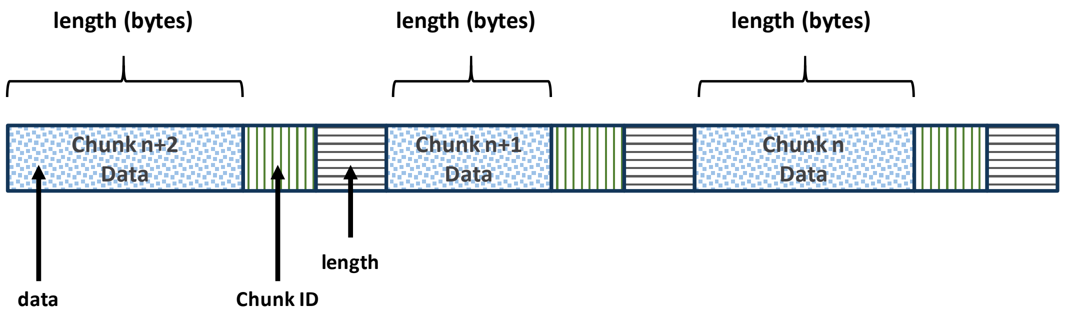


Figure 2: Layout of Chunk Data within a Data Block

In MISB ST 1608, metadata follows the Motion Imagery within a data payload. This organization maintains a strict binding between the imagery and its associated metadata.

Requirement	
ST 1608-02	Metadata associated with a Motion Imagery frame shall follow the Motion Imagery data within the same payload.

Figure 3 shows the organization of Image Data followed by an Image Chunk ID and Image Chunk Length, followed by the Metadata Data and its Metadata Chuck ID and Metadata Chunk Length. The two Chunk IDs are reused in the successive data blocks that follow for Image 2, 3, through K and Metadata 2, 3, through K.

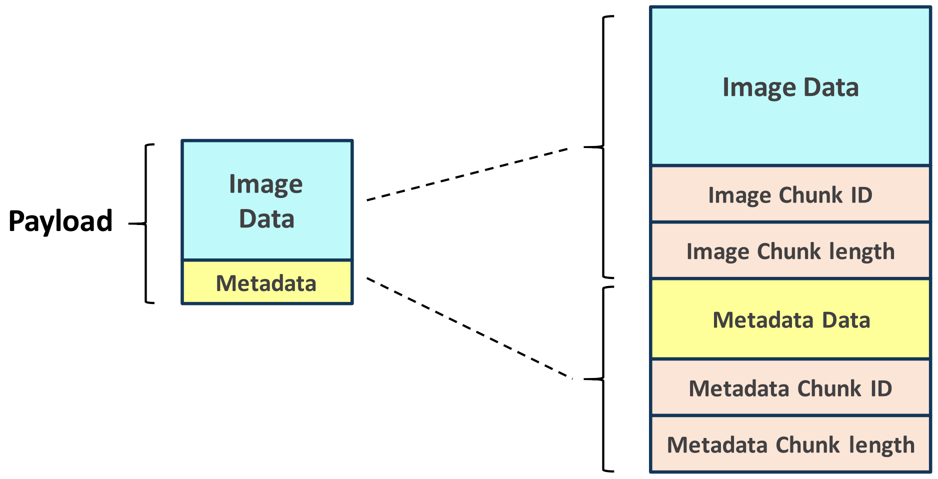
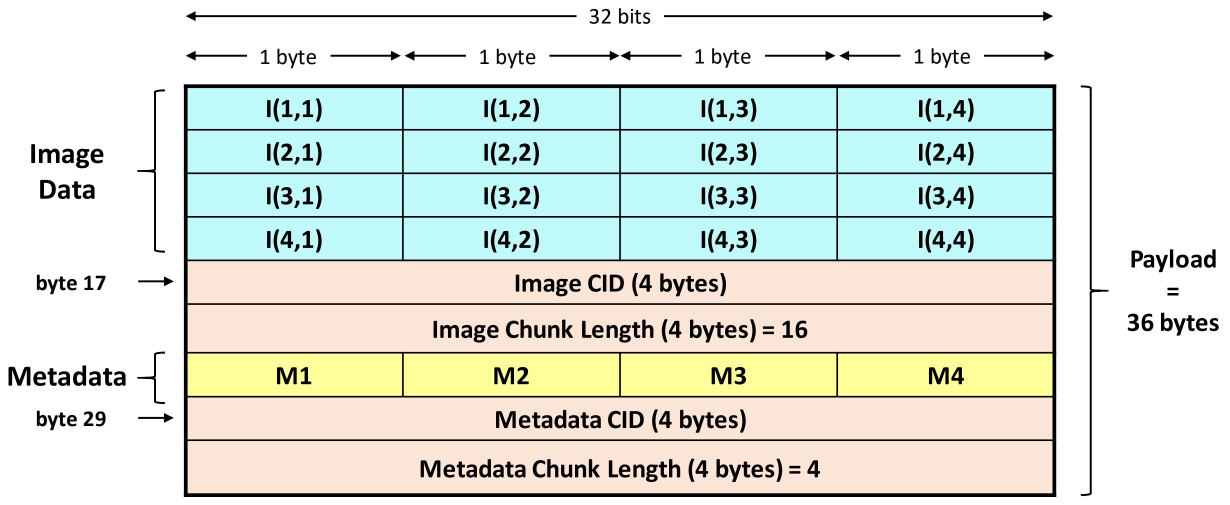


Figure 3: Data Organization within Payload using Extended Chunk Mode

In the extended Chunk mode, the length in bytes of all chunks within a payload are summed and stored in the Data Trailer Packet in the field called chunk_data_payload_length, which must be a multiple of 4 bytes. This length is needed to find the CID associated with the last chunk. Once this CID is found, the length for its chunk can be used to locate the CID for the next chunk of data, and so on.

Example of Chunk Parsing

Figure 4 illustrates an example of chunk data where there are 16 bytes of image data in an image chunk, and 4 bytes of metadata in the metadata chunk. Accounting for the four-byte CID and the four-byte length for each chunk, the total payload in the data block is 36 bytes. This value is assigned to the *chunk_data_payload_length* inserted into the Data Trailer Packet.



Chunk_data_payload_length = Σ payload bytes
= 16 (image) + 8 (CID/Length) + 4 (metadata) + 8 (CID/Length)
= 36 bytes

Metadata CID: Chunk_data_payload_length – 8 (chunk length and CID) = 36 – 8 = 28, or byte 29
Image CID: 28 – metadata chunk – 8 = 28 – 4 – 8 = 16, or byte 17

Figure 4: Example – Calculating CID position for each Chunk

To parse the two chunks, the position for the metadata CID is first found by subtracting the metadata length and metadata CID, which are 8 bytes in total, from chunk_data_payload_length. The metadata CID is found after byte 36 - 8 = 28, or at byte 29. The position for the image CID is computed as: chunk_data_payload_length minus length for the metadata minus 8 bytes for the metadata CID and length minus 8 bytes for the image CID and length, which is 36 - 8 - 4 - 8 = 16 bytes. The image CID then is found after byte 16 at byte 17.

The length value for each chunk can be found after its corresponding CID, and used to parse each its respective chunk data. This process permits payload data to be added without knowing the quantity of data apriori.

6.1 Motion Imagery

Within the Data Leader Packet, a 16-bit payload type defines the type of data in the data block. Data Payload Packets transport the actual image and metadata information to a GVSP receiver.

6.1.1 Class 0 Motion Imagery

A *payload_type* = 0x4001 supports a payload type of "Image" with extended chunk data, which is for non-compressed image data followed by other data, such as metadata; this payload type is called *image extended chunk data*. In this mode, the Data Leader Packet allows selection of scan type, pixel format, image dimensions, etc. Various pixel layouts and formats for imagery are defined in Section 26 of the GigE Vision specification.

Requirement	
ST 1608-03	When inserting metadata along with Class 0 Motion Imagery into a GigE Vision interface, the Data Leader Packet payload_type shall be 0x4001 (payload_type = 0x4001).

6.1.2 Class 2 Motion Imagery – H.264

A *payload_type* = 0x4008 supports a payload type of "H.264" with extended chunk data, which is for H.264-compressed image data followed by other data, such as metadata; this payload type is called *image extended chunk data*. In this mode, the Data Leader Packet allows selection of scan type, packetization mode, profile/level, etc.

Requirement	
ST 1608-04	When inserting metadata along with H.264/AVC Class 2 Motion Imagery into a GigE Vision interface, the Data Leader Packet payload_type shall be 0x4008 (payload_type = 0x4008).

6.1.1 Class 2 Motion Imagery – JPEG 2000

A *payload_type* = 0x4007 supports a payload type of "JPEG 2000" with extended chunk data, which is for J2K-compressed image data followed by other data, such as metadata; this payload type is called *image extended chunk data*. In this mode, the Data Leader Packet allows selection of scan type, color space, data format, etc.

Requirement	
ST 1608-05	When inserting metadata along with JPEG 2000 Class 2 Motion Imagery into a GigE Vision interface, the Data Leader Packet payload_type shall be 0x4007 (payload_type = 0x4007).

6.2 KLV Metadata

KLV is an acronym for Key-Length-Value. Developed by SMPTE, KLV is a self-describing encoding structure. The KLV triplet encodes a data value (V), along with a count of bytes needed to support the data value (L), and an identifier (K), as one block of binary data. The Key is a reserved set of 16 bytes used as an index into a dictionary of registered Keys. The Length is BER-encoded (Basic Encoding Rules), and specifies the number of bytes needed to describe the value. The Value is a block of bytes encoded using the rules specified for the given Key. The Value can be a single numeric value (i.e. integer or float), string, a binary value, or a group of items (such as a Local Set) of any length. See the Motion Imagery Handbook [3] for more details.

The GigE Vision specification does not define how to encode metadata, nor how to resolve different data or metadata types – only where to place non-imagery data. This standard identifies the carriage of KLV metadata within the GigE Vision defined data space using the extended chunk mode.

With a *payload_type* = 0x4001, 0x4007, or 0x4008, metadata can be inserted following image data. In the extended chunk mode, the image data represents one chunk, and metadata a second chunk. Each chunk requires specifying two 32-bit fields: its identifier (i.e. the Chunk ID), and the length of the chunk (i.e. the Chunk length*)*.

KLV metadata is organized as a linear array of KLV items, with no discontinuities between items as shown in Figure 5. Here KLV Metadata Set 1 is the first KLV item followed by KLV Metadata Set 2, and so forth. The KLV may a KLV Local Set, Pack structure, or other allowed KLV construct.

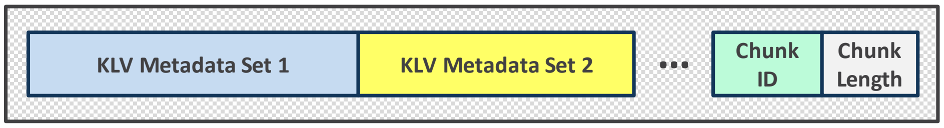


Figure 5: KLV Metadata Organized within Chunk Data Space

The Chunk ID (CID) is a four-byte identifier for the chunk data as KLV metadata, and the Chunk Length, another four-byte value, is the sum of all bytes beginning with the first Key to the Chunk ID. Data within the chunk data space is required to be a multiple of 4 bytes, so padding of the data may be necessary. Both the CID and Chunk Length must be a multiple of four (4).

The chunk data description for KLV metadata is summarized in Table 1. Position represents the first byte of the chunk identified by the CID.

Table 1: Chunk Structure Description for KLV Metadata

Position	Format	Description
0	KLV [K Bytes]	The KLV data transported by the chunk. This section must be a multiple of 4 bytes, and if not, padded such that K is a multiple of 4 bytes. This ensures CID and length fields are 32-bit aligned within the chunk. Data is big-endian.
K	CID [4 Bytes]	Chunk IDentifier. The CID in big-endian.
K+4	Chunk Length [4 Bytes]	The length of the data in bytes, a multiple of 4, big-endian

Table 2 shows an example of chunk data expressed in a GVSP data block for the KLV organized as shown in Figure 5.

Table 2: Example GVSP Data Block

	Byte 0	Byte 1	Byte 2	Byte 3	…	Byte 1919	Description
Image chunk	0x0E	0x15	0x64	0x45	…	0x64	Image data + 0 padding bytes (big-endian)
	0x32	0x35	0x48	0x75	…	0xFF	
	…	…	…	…	…	…	
	0x72	0x39	0x25	0x30	…	0x04	
	0x00	0x00	0x01	0x00			Chunk ID for image data (big-endian)
	0x00	0x1F	0xA4	0x00			Chunk Length = 2,073,600 bytes (big-endian)
Metadata chunk	0x06	0x0E	0x2B	0x34	…	…	KLV metadata + 2 padding bytes (big-endian)
	0x3C	0x1D	0x0F	0x34			Chunk ID for metadata (big-endian)
	0x00	0x00	0x01	0x90			Chunk Length = 400 bytes (big-endian)

Table 2 assumes the following parameters:

- Image pixel density: 1920x1080
  - Mono8 gray values
  - Chunk ID: 0x00001000
  - Chunk Length: 2,073,600 bytes (0x001FA400)
- KLV Metadata
  - MISB ST 1507 Sensor Timing Local Set (see Section. 6.2.2 below)
  - MISB ST 0902 Metadata Set
  - Chunk ID: 0x3C1D0F34
  - Chunk Length: 43 bytes (Sensor Timing Local Set) + 355 bytes (MISB ST 0902) + 2 bytes (padding) = 400 bytes (0x0000190)

Other organizations of KLV, as shown in Figure 6, can be employed for metadata carriage provided there are no discontinuous breaks in the data represented. If the metadata is of the same type (e.g. KLV), then the value for the Chunk ID can remain the same. The Chunk Length will be a function of the number of bytes representing the KLV within its chunk.

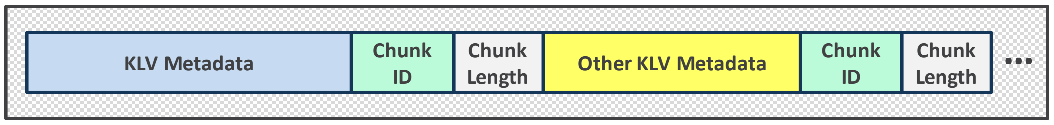


Figure 6: Alternative Organization of KLV Metadata within Chunk Data Space

6.2.1 Inserting KLV into the Chunk Data Space

Requirement	
ST 1608-06	KLV metadata inserted into the chunk data space shall be in big-endian byte order in accordance with MISB ST 0107 [2].

6.2.2 MISB ST 1507 Motion Imagery Sensor Timing Metadata

MISB ST 1507 [3] provides a KLV Sensor Timing Local Set intended to precisely define timing information in Motion Imagery. In its most basic instantiation, it provides an Enhanced Precision Time Stamp, a Region List and a CRC for the set; these are the required elements of MISB ST 1507 for MISB ST 1608.

6.2.2.1 Enhanced Precision Time Stamp

The GigE Vision Specification provides (optional) support for a GigE Vision timestamp that is derived from IEEE 1588-2008. Although the GigE Vision timestamp and the MISB Enhanced Precision Time Stamp share the same epoch, this standard requires the MISP Time System as defined in the Motion Imagery Handbook and directed by MISB ST 1603 [4].

There are several reasons the Enhanced Precision Time Stamp is to be used in lieu of or in addition to the GigE Vision timestamp:

1. The GigE Vision timestamp is optional in the GigE Vision Specification. Some cameras may not provide it, or may generate it from an internal clock.
2. The GigE Vision timestamp as indicated in GigE Vision represents time when a block of data is generated. There is no information provided on how the timestamp aligns to a specific instance within an image, for example, start of exposure, end of exposure, etc.

The Enhanced Precision Time Stamp, defined in MISB ST 1603, is a KLV pack structure composed of two components: a 64-bit Nano-Second counter referenced to the MISP Time System epoch (1970-01-01T00:00:00.0Z, see MISB ST 0603 [5]), and the Time Transfer Local Set, which provides metadata to validate and correct timestamps, and a leap second offset to adjust International Atomic Time (TAI) to Universal Coordinated Time (UTC).

6.2.2.2 Inserting an Enhanced Precision Time Stamp into the Chunk Data Space

An Enhanced Precision Time Stamp is required for each frame of Motion Imagery. This information, which is encoded within the MISB ST 1507 Sensor Timing Local Set, is inserted into the chunk data space prior to other MISB KLV data.

Figure 7 shows an example of a MISB ST 1507 Sensor Timing Local Set followed by a MISB ST 0601 Local Set specifying the MISB ST 0902 minimum metadata set elements. Note the Enhanced Precision Time Stamp is an element of the MISB ST 1507 Sensor Timing Local Set. In turn, the Enhanced Precision Time Stamp is itself a combination of a 64-bit Nano-Second counter and the Time Transfer Local Set.

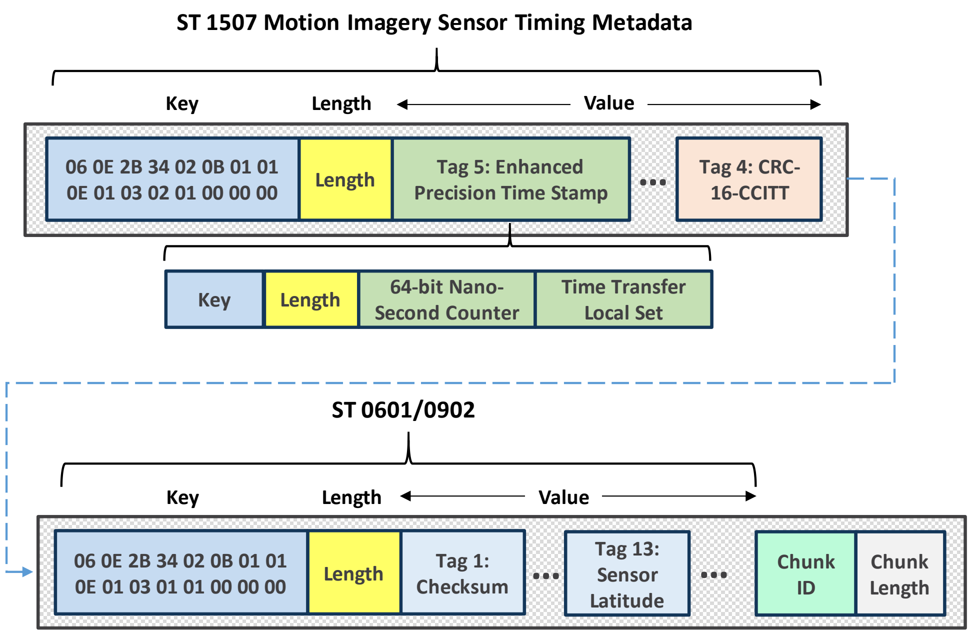

ST 1507 Motion Imagery Sensor Timing Metadata
Key
Length
Value
06 0E 2B 34 02 0B 01 01 0E 01 03 02 01 00 00 00
Length
Tag 5: Enhanced Precision Time Stamp
Tag 4: CRC-16-CCITT
Key
Length
64-bit Nano-Second Counter
Time Transfer Local Set
ST 0601/0902
Key
Length
Value
06 0E 2B 34 02 0B 01 01 0E 01 03 01 01 00 00 00
Length
Tag 1: Checksum
Tag 13: Sensor Latitude
Chunk ID
Chunk Length

Figure 7: Example KLV Metadata in Chunk Data Space

Requirement(s)	
ST 1608-07	When implementing GigE Vision, an Enhanced Precision Time Stamp in accordance with MISB ST 1603 shall be present with every Motion Imagery frame within the chunk data space.
ST 1608-08	The Enhanced Precision Time Stamp encoded within a MISB ST 1507 Sensor Timing Local Set shall be the first KLV item present in the chunk data space.

7 Carriage of Metadata Only

This section defines the GigE Vison payload type for transport of metadata only (i.e. without imagery).

A *payload_type* = 0x0004 supports a payload type of “Chunk Data”

Requirement	
ST 1608-09	When inserting metadata-only into a GigE Vision interface, the Data Leader Packet payload_type shall be 0x0004 (payload_type = 0x0004).

The Chunk Data Leader Packet, Payload Packet, and Trailer Packet are defined in GigE Vision Section 25.5. In this mode, there is no primary payload type, so the first chunk can be metadata. The rules for formatting and inserting metadata are the same as described in Section 6.2 above.

8 Camera Description File – Informative

The GenICam Standard [6] is a generic Application Programming Interface (API), which allows for control and operation of a GigE Vision compliant device, typically a camera. At the core of GenICam is a device XML description file. The application retrieves the device's device XML description file to configure the interface and control the device.

A camera is described by means of a camera description file, which contains the required information to map a camera's features to its registers. The camera description file containing a set of nodes with each node having a type and a unique name, or Name attribute. The Name must be unique within the camera description file.

Each *Name* lives inside a name space identified by the *NameSpace* attribute of the Node, which can have two possible values: *Custom* or *Standard*. If it is *Custom*, any name can be used if it is unique within the camera description file. If it is *Standard*, it must come from the standard feature name lists as defined in the GenICam Standard Features Naming Convention (SFNC) for GigE Vision.

Configuration of a device is controlled through several bootstrap registers; these registers when used should be provided in the XML description file (see Section 28 in [1]). A First URL (register address 0x200) and a Second URL (register address 0x400) are used to indicate the location of the XML description file.

8.1.1 Example Node

```
<RegisterDescription
     <Float Name="ChunkExposureTime" NameSpace="Standard">
       <ToolTip>Exposure Time of the image</ToolTip>
       <Description>Returns the exposure time used to capture the image</Description>
       <DisplayName>Chunk Exposure Time</DisplayName>
       <Visibility>Guru</Visibility>
       <pValue>chunkExposureTimeReg</pValue>
     </Float>
     <IntReg Name="chunkExposureTimeReg">
       <Address>0</Address>
       <Length>4</Length>
       <AccessMode>RO</AccessMode>
       <pPort>chunkPort</pPort>
     </IntReg>

     <Port Name="chunkPort">
       <ChunkID>CD000001</ChunkID>
     </Port>
</RegisterDescription>
```

This example node describes ChunkExposureTime in a data stream such as:

Name	Size (In bytes)	Value
Image	640x480	Raw image data
ChunkID	4	0xCD000000 (4 bytes)
ChunkSize	4	307200 bytes

ChunkExposureTime	4	The time the frame was exposed)
ChunkID	4	0xCD000001 (4 bytes)
ChunkSize	4	4 bytes

9 KLV Metadata Description File

To facilitate parsing MISB KLV metadata within the chunk data space, the node name "MISB_KLV" is defined within the *Custom* name space. This name is associated with a specific <ChunkID> element, used to identify KLV data in a chunk buffer. The <ChunkID> element is mapped to a virtual port (i.e. KLVchunkPort), which does not give access to a real device, but rather to the chunk of data residing in memory. The <Address> element defines the first position in the chunk buffer holding the metadata. A default Length of zero (0) is defined for the <Length> element. See the GenICam Standard for XML names and types.

Example MISB_Metadata.xml file (informative):

```
<RegisterDescription
     <Register Name="MISB_KLV" NameSpace="Custom">
       <ToolTip>MISB KLV metadata</ToolTip>
       <Description>KLV Metadata related to frame of Motion Imagery</Description>
       <DisplayName>MISB Metadata</DisplayName>
       <Visibility>Expert</Visibility>
       <Address>0</Address>
       <Length>0</Length>
       <AccessMode>RW</AccessMode>
       <pPort>KLVchunkPort</pPort>
       <Endianess>BigEndian</Endianess>
     </Register>

     <Port Name="KLVchunkPort">
       <ChunkID>cd000001</ChunkID>
     </Port>
</RegisterDescription>
```

Each chunk is identified by its unique Chunk ID (a four-byte value) which maps it to the corresponding port node in the device's XML description file. To address the data in the chunk the implementation must know the position (offset) of the chunk in the buffer and its size. The structure of the chunk data in the buffer is technology specific. Each chunk when read into buffer memory is referenced to a starting address within the buffer with KLV data offset by the given address.

Requirement	
ST 1608-10	A device XML description file defining a unique Chunk ID (CID) for KLV metadata shall be included in the XML description file indicated by the GigE Vision Bootstrap First URL register value.